REMARKS

Claims 1 and 3-70 are all the claims presently pending in the application. Claims 1, 45, 53, and 64 are amended to more clearly define the invention. Claims 1, 45, 53, 64-66, and 70 are independent.

Applicants also note that, notwithstanding any claim amendments herein or later during prosecution, Applicants' intent is to encompass equivalents of all claim elements.

Entry of this §1.116 Amendment is proper. Since the Amendments above narrow the issues for appeal and since such features and their distinctions over the prior art of record were discussed earlier, such amendments do not raise a new issue requiring a further search and/or consideration by the Examiner. As such, entry of this Amendment is believed proper and Applicants earnestly solicit entry. No new matter has been added.

Applicants gratefully acknowledge the Examiner's indication that claims 65 and 66 are allowed. However, Applicants respectfully submit that all of the claims are allowable.

Claims 1-3, 5, 7-13, 17-64, and 67-70 stand rejected under 35 U.S.C. § 102(b) as being anticipated by the Template Software product line reference. Claims 4, 6, and 14-16 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over the Template Software product line reference in view of the UML Schema reference.

These rejections are respectfully traversed in the following discussion.

I. THE CLAIMED INVENTION

A first exemplary embodiment of the claimed invention, as defined by, for example, independent claim 1 is directed to a method for detecting an error in an interaction between a plurality of software systems, that includes providing information about at least one of at least first and second software systems, and a mapping between at least a portion of the at least first and second software systems, automatically generating a check based upon the information and the mapping, inserting the check into at least one of the first and second software systems, and examining the at least one of the first and second software systems and the mapping to determine

an error in an interaction between the at least first and second software systems based upon the check.

A second exemplary embodiment of the claimed invention, as defined by, for example, independent claim 70 is directed to a method for detecting an error in an interaction between a plurality of software systems, that includes providing information about at least one of at least first and second software systems, and a mapping between at least a portion of the at least first and second software systems, and statically examining the at least one of the first and second software systems and the mapping to determine an error in an interaction between the at least first and second software systems.

Conventional application programming frameworks involve databases that have integrity constraints while the applications which interact with those databases do not have integrity constraints. Thus, an application which attempts to interact with that database may cause an error because of an integrity constraint violation.

To address these issues, conventional programming frameworks, such as the Template Software product line reference framework, rely upon a user to insert code to check for errors. In order to generate correct checks, these frameworks rely upon the user to have knowledge about the integrity constraints and the relationships between the application and the database.

In stark contrast, the first exemplary embodiment of the present invention provides a method which includes generating a check based upon the information and the mapping. In this manner, the method does not rely upon a user to generate the check and the user is not required to perform this function. This significantly reduces the workload of the user.

This method and system is also advantageous in that conventional frameworks have required users to constantly update and monitor the information about the software systems. If this information changes, then the user must perform the function of again generating a check in accordance with the change. In stark contrast, the method and system of the invention automatically generates the check without requiring the user to generate the check.

Futher, conventional systems and methods which have statically examined applications have not been able to detect errors in interactions between software systems based on a mapping between the two systems.

In stark contrast, the second exemplary embodiment of the present invention provides a method which statically examines the at least one of the first and second software systems and the mapping to determine an error in an interaction between the at least first and second software systems. In this manner, the software systems do not require dynamic examination of these systems to determine errors as have been conventionally required. In this manner, the mapping between the two software systems may be accounted for during the static examination in addition to the information (such as, for example, integrity constraints) when detecting conditions under which errors will always occur without executing the software systems.

II. THE PRIOR ART REJECTIONS

A. The Response to Arguments

In the "Response to Arguments" section of the March 20, 2007, Office Action, the Examiner agrees that the Applicants' "statements about a user inserts code to check for errors [in the applied reference] is correct," but points out that "[t]he current claim limitations do not appear to distinguish who or what inserts the invention checks. It is the Examiner's believe (sic) that the two limitations if clearly claimed will distinguish the invention."

In this regard, this Amendment amends independent claims 1, 45, 53, and 64 to clarify that the claimed invention automatically generates a check based upon the information and the mapping. This clarifies that it is the claimed invention which generates the checks and that it is not a user that generates the checks.

With respect to independent claim 70, the Examiner alleges that "The prior argument that the mapping is 'statically' examined is not able to be determined distinct." Applicant respectfully submits that the term "statically" is a term which is well understood by those of ordinary skill in the art and refers to a state where there software system which is being examined is not operating. A "static" examination is the opposite of a "dynamic" examination because, a "static" examination occurs when the system being examined is not operating and a "dynamic" examination occurs when the system being examined is operating.

For example, conventional examination systems typically are only able to dynamically determine an error in interaction between two systems. This means that the conventional

systems require both systems to be operating in order to determine whether there is an interaction error between them.

Further, while some conventional examination systems have been able to statically examine systems, these conventional systems have not been able to detect errors in operation between software systems based upon a mapping between the two systems.

None of the applied references teaches or suggests statically examining the at least one of the first and second software systems and the mapping to determine an error in an interaction between the at least first and second software systems.

B. The Template Software product line reference

Regarding the rejection of claims 1-3, 5, 7-13, 17-64, and 67-70, the Examiner alleges that the Template Software product line reference teaches the claimed invention. Applicants submit, however, that there are elements of the claimed invention which are neither taught nor suggested by the Template Software product line reference.

None of the applied references teaches or suggests the features of the claimed invention including a method and system which automatically generates a check based upon the information and the mapping, or a method and system which statically examines the at least one of the first and second software systems and the mapping. As explained above, these features are important for obviating the necessity of requiring a user to generate and provide such a check. Rather, the method and system automates the generation of the check.

The Template Software product line reference merely provides a framework within which a user may insert code to check for errors. This requires the user to generate the check in order to insert the check. COM 4-21 and SNAP 3-44 to 3-50 of the Template Software product line reference disclose how the user may provide functions (which may throw errors). Clearly, the Template Software product line reference does not teach or suggest a method and system that generates a check based upon the information and mapping.

Indeed, the Template Software product line reference does not disclose a method and system which generates a check at all, let alone a method and system that generates a check based upon the information and mapping. Rather, the filter functions that appear to be disclosed

by the Template Software product line reference at, for example, COM 5-8, 7-3, and 8-2, are mechanisms by which a user (e.g., a programmer) may provide the error checking code.

In stark contrast, the present invention provides method and system which automatically generates a check based upon the information and the mapping. In this manner, the invention may transparently generate the check based upon the provided information, thus reducing the burden on the programmer and making the software systems more maintainable.

Clearly, the Template Software product line reference does not teach or suggest the features of the claimed invention including a method and system which automatically generates a check based upon the information and the mapping or a method and system which statically examines the at least one of the first and second software systems and the mapping.

Therefore, the Template Software product line reference does not teach or suggest each and every element of the claimed invention and the Examiner is respectfully requested to withdraw this rejection of claims 1-3, 5, 7-13, 17-64, and 67-70.

C. The Template Software product line reference in view of the XML Schema reference

Regarding the rejection of claims 4, 6, and 14-16, the Examiner alleges that the XML Schema reference would have been combined with the Template Software product line reference to form the claimed invention. Applicants submit, however, that these references would not have been combined and, even if combined, the combination would not teach or suggest each and every element of the claimed invention.

None of the applied references teaches or suggests the features of the claimed invention including a method and system which automatically generates a check based upon the information and the mapping, or a method and system which statically examines the at least one of the first and second software systems and the mapping. As explained above, these features are important for obviating the necessity of requiring a user to generate and provide such a check. Rather, the method and system automates the generation of the check.

As explained above, the Template Software product line reference does not teach or suggest these features.

The XML Schema reference does not remedy the deficiencies of the Template Software product line reference.

Rather, the XML Schema reference discloses an entirely different type of basis upon which a check may be generated. The check generator of the XML Schema reference merely understands the different semantics and integrity constraints that are allowed by the XML schema and generates checks based upon that information.

Clearly, the XML Schema reference does not teach or suggest a method and system which automatically generates a check based upon the information and the mapping.

Therefore, none of the applied references teaches or suggesst each and every element of the claimed invention and the Examiner is respectfully requested to withdraw this rejection of claims 4, 6, and 14-16.

III. FORMAL MATTERS AND CONCLUSION

In view of the foregoing amendments and remarks, Applicants respectfully submit that claims 1 and 3-70, all the claims presently pending in the Application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the Application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,

Date: May 16, 2007

James E. Howard
Registration No. 39,715

McGinn Intellectual Property Law Group, PLLC
8321 Old Courthouse Rd., Suite 200
Vienna, Virginia 22182
(703) 761-4100
Customer No. 48150